```
TITLE "ARM_HPI";

% Program to translate from ARM IO to cLead HPI IO %
% Version X10 DRV_FLAG   at HQ=F only   %

CONSTANT DSP_BASE0 = H"350";      % Same as API_0 in Odessa plus A20 high                 %


SUBDESIGN Arm_HPI
(
        CLK_ARM   :    INPUT;       % Clock signal from the ARM  40MHz
%

     DRV_FLAG        : OUTPUT;      %Flag to monitor state of driving during read         %


     NRW        :    INPUT;         % ARM WR ( Low = Read, High = Write )                 %

    DLYNMREQ    :    INPUT;         % Delayed memmory request used durring R or W         %


     EXT_INT    :    OUTPUT;        % ARM interupt request to the processor "active low"  %
                                    %  connected to  NFIRQ in ARM                         %
     NWAIT      : OUTPUT;           % ARM signal used during read cycles                  %

     SA[14..1]  :    INPUT;          % ARM address bus, only enough for I/O               %
                                     % Connect 3..14 to 16..27 in ARM and 0,1,2 to 0,1,2  %


     SD[15..0] :    BIDIR;          % ARM BIDIRECTIONAL BUS                               %


     DSP0CS     :    OUTPUT;        % HPI level selects for  DSP                          %

     DSP0IRQ    :    INPUT;         % The Interupt request signal from the DSP            %


     HD[7..0]   :    BIDIR;         % The data bus for the host port into the DSPs        %
     HCTL0      :    OUTPUT;        % Control bit 0 (really A1)                           %
     HCTL1      :    OUTPUT;        % Control bit 1 (really A2)                           %
     HBIL       :    OUTPUT;        % Signal about which byte is being transferred        %
     HDS1       :    OUTPUT;        % A data strobe for the interface                     %
     HDS2       :    OUTPUT;        % Same as above, but only one can be used             %

   % HAS Pull up in Hardware %

     HR_W       :    OUTPUT;        % Read is active high, Write is active low            %
     HRDY       :    INPUT;         % When LOW, DSP wants more time                       %



)
VARIABLE
    drv_var          :    NODE;
    drv_flag         :    NODE;
    NWAIT            :    NODE;          %  Wait state is syncronous to ARM CLK           %
    NWAITNSS         :    NODE;          %  Wait state is asyncronous to ARM CLK          %
    NWAITS1          :    NODE;          %  Wait state is in state machine                %
    VS               :    NODE;          % Variable used for NWAIT                        %


     HDS1             :    DFF;               % Data strobe is syncronous, so its registered%

     sdnode[15..0]    :    TRI_STATE_NODE;  % Tristates for the ARM data bus to read       %
    hdnode[7..0]     :    TRI_STATE_NODE;  % Tristate bus for the DSP data side           %
    add_is_hpi       :    NODE;            % Indication that the address matches ours     %
    io_is_hpi        :    NODE;            % address match AND its an IO cycle            %
    hq[3..0]         :    DFFE;            % A counter to build a HPI state machine       %
    lo_byte[7..0]    :    DFFE;            % Latch to hold high byet on HPI read          %
    low_latch        :    DFFE;            % Node to latch the low byte on read           %
    hpi_read         :    NODE;            % Decode of an active ARM read of the HPI      %
    hpi_write        :    NODE;            % Decode of an active ARM write to an HPI      %
    count_clear      :    NODE;            % Flag that will enable counter when in s1     %
    ss: MACHINE WITH STATES (s0, s1, s2, s3, s4, s5);

BEGIN
%                                                                                        %
%  First we need a few general terms defined/calculated                                 %
%         (here a check is made of the ARM target/source addresses & strobes)           %


add_is_hpi = SA[14..3] == DSP_BASE0;              % ARM may be for DSP (3500000 h)     %


io_is_hpi = add_is_hpi & !DLYNMREQ;              % ARM is really a Write cycle intended for
HPI  %


hpi_read  = add_is_hpi & !NRW;          % read                           %
hpi_write = add_is_hpi &  NRW;           % write                              %



NWAITNSS = add_is_hpi & !DLYNMREQ & VS;
NWAIT  = !NWAITNSS & !NWAITS1;


DRV_FLAG = drv_var & hpi_read;

ss.clk = CLK_ARM;
ss.reset = !add_is_hpi;

CASE ss IS
    WHEN s0 =>
        count_clear = GND;
        DSP0CS = VCC;
        VS = VCC;
        drv_var = GND;
        IF io_is_hpi THEN
        ss = s1;
        END IF;

    WHEN s1 =>
        count_clear = VCC;
        VS = GND;
        drv_var =  hq[3..0] == H"F";

        NWAITS1 = hq[3..0] == H"0" & !VS
                # hq[3..0] == H"1"  & !VS
                # hq[3..0] == H"2"  & !VS
                # hq[3..0] == H"3"  & !VS
                # hq[3..0] == H"4"  & !VS
                # hq[3..0] == H"5"  & !VS
                # hq[3..0] == H"6"  & !VS
                # hq[3..0] == H"7"  & !VS
                # hq[3..0] == H"8"  & !VS
                # hq[3..0] == H"9"  & !VS
                # hq[3..0] == H"A"  & !VS
                # hq[3..0] == H"B"  & !VS
                # hq[3..0] == H"C"  & !VS
                # hq[3..0] == H"D"  & !VS
                # hq[3..0] == H"E"  & !VS;
                !DSP0CS = (SA[14..3]== DSP_BASE0);

          IF (hq[3..0] == H"F") THEN
              ss = s2;
          END IF;

      WHEN s2 =>
          DSP0CS = VCC;
          VS = GND;
          drv_var = GND;
          ss = s3;

      WHEN s3 =>
          DSP0CS = VCC;
          VS = GND;
          drv_var = GND;
          ss = s4;
      WHEN s4 =>
          DSP0CS = VCC;
          VS = GND;
          drv_var = GND;
          ss = s5;
      WHEN s5 =>
          DSP0CS = VCC;
          VS = GND;
          drv_var = GND;
          ss = s0;


END CASE;

%                                                                          %
% Host port interface stuff                                                %
%                                                                          %


hq[3..0].clk  = CLK_ARM;                      % HPI runs at a ARM speed            %
hq[3..0].clrn = count_clear;                    % Counter ony runs when HPI is active %

hq[3].d = hq[3] $ hq[2] & hq[1] & hq[0]        % Counter is used be HPI state system %
        # hq[3];                               %  to generate acceptable waveforms   %
hq[2].d = hq[2] $ hq[1] & hq[0]                %  into the DSP's HPI ports           %
        # hq[3] & hq[2] & hq[1] & hq[0];       %                                     %
hq[1].d = hq[1] $ hq[0]                        %  the counter hits the top and       %
        # hq[3] & hq[2] & hq[1] & hq[0];       %   pegs there, it does not wrap.     %
hq[0].d = !hq[0]                               %                                     %
        # hq[3] & hq[2] & hq[1] & hq[0];       %                                     %
hq[3..0].ena = HRDY;                           % Pause State machine if DSP not ready%

HCTL0 = SA[1];             % The HPI control bits are really two address bits - to sixteen  %
HCTL1 = SA[2];             %   bit words - so forward these low order word addresses        %
HR_W  = !NRW;              % ARM Write is passed to the HPI as a write signal - not a strobe%
HDS1.d  = !hq[2] & !hq[1] & !hq[0]    % The data strobe is 40ns wide (minimum) and         %
        #  hq[2] & !hq[1] &  hq[0]    %  is active twice during the count cycle.  As       %
        #  hq[2] &  hq[1] & !hq[0]    %  only one data strobe can be active, the HDS2      %
        #  hq[2] &  hq[1] &  hq[0];   %  signal is permanently disabled.                   %
```

```
HDS1.clk = CLK_ARM;                     %                                                  %
HDS2 = VCC;                              %                                                  %
HBIL = hq[3];                            % Low for first byte, high for second              %

% Wait state for the ARM during read or write cycle  %



low_latch.clk = CLK_ARM;                            % A short pulse that will latch the     %
low_latch.d = !hq[3] & hq[2] & !hq[1] & hq[0];      %  first bye of a word read operation   %

lo_byte[7..0].clk = CLK_ARM;                   % The first byte of a word transfer          %
lo_byte[7..0].ena = low_latch;                  %  from the DSP's HPI to the ISA bus is      %
lo_byte[7].d = hd[7];                           %  held in a register so that it will be     %
lo_byte[6].d = hd[6];                           %  availible when the second byte of the     %
lo_byte[5].d = hd[5];                           %  transfer is ready.                        %
lo_byte[4].d = hd[4];                           %                                            %
lo_byte[3].d = hd[3];                           %                                            %
lo_byte[2].d = hd[2];                           %                                            %
lo_byte[1].d = hd[1];                           %                                            %
lo_byte[0].d = hd[0];                           %                                            %

sdnode[0] = TRI (hd[0], hpi_read & drv_var);          % Sixteen data bits are placed on the
ARM   %
sdnode[1] = TRI (hd[1], hpi_read & drv_var);          %  bus during read.
      %
sdnode[2] = TRI (hd[2], hpi_read & drv_var);          %
      %
sdnode[3] = TRI (hd[3], hpi_read & drv_var);          %
      %
sdnode[4] = TRI (hd[4], hpi_read & drv_var);          %
      %
sdnode[5] = TRI (hd[5], hpi_read & drv_var);          %
      %
sdnode[6] = TRI (hd[6], hpi_read & drv_var);          %
      %
sdnode[7] = TRI (hd[7], hpi_read & drv_var);          %
      %
sdnode[8]  = TRI (lo_byte[0], hpi_read & drv_var);    %
      %
sdnode[9]  = TRI (lo_byte[1], hpi_read & drv_var);    %
      %
sdnode[10] = TRI (lo_byte[2], hpi_read & drv_var);    %
      %
sdnode[11] = TRI (lo_byte[3], hpi_read & drv_var);    %
      %
sdnode[12] = TRI (lo_byte[4], hpi_read & drv_var);    %
      %
sdnode[13] = TRI (lo_byte[5], hpi_read & drv_var);    %
      %
sdnode[14] = TRI (lo_byte[6], hpi_read & drv_var);    %
      %
sdnode[15] = TRI (lo_byte[7], hpi_read & drv_var);    %
      %

hdnode[0] = TRI (SD[0],  hpi_write &  HBIL); % This little bbock puts out the data into  %
hdnode[1] = TRI (SD[1],  hpi_write &  HBIL); %  the DSPs one byte at a time.              %
hdnode[2] = TRI (SD[2],  hpi_write &  HBIL); %                                            %
hdnode[3] = TRI (SD[3],  hpi_write &  HBIL); %                                            %
hdnode[4] = TRI (SD[4],  hpi_write &  HBIL); %                                            %
hdnode[5] = TRI (SD[5],  hpi_write &  HBIL); %                                            %
hdnode[6] = TRI (SD[6],  hpi_write &  HBIL); %                                            %
hdnode[7] = TRI (SD[7],  hpi_write &  HBIL); %                                            %
hdnode[0] = TRI (SD[8],  hpi_write & !HBIL); %                                            %
```

```
hdnode[1] = TRI (SD[9],  hpi_write & !HBIL); %                                                  %
hdnode[2] = TRI (SD[10], hpi_write & !HBIL); %                                                  %
hdnode[3] = TRI (SD[11], hpi_write & !HBIL); %                                                  %
hdnode[4] = TRI (SD[12], hpi_write & !HBIL); %                                                  %
hdnode[5] = TRI (SD[13], hpi_write & !HBIL); %                                                  %
hdnode[6] = TRI (SD[14], hpi_write & !HBIL); %                                                  %
hdnode[7] = TRI (SD[15], hpi_write & !HBIL); %                                                  %

     % Pretty straight forward, addresses  %
                                                       %  are compared against a DSP's range %
                                                       %  and qualified to ensure that the   %
                                                       %  ARM bus is in an active IO cycle.  %
                                                       %  Since these are active low, the    %
                                                       %  signal is assigned with an         %

% If the DSP interupt is actice, then the ARM bus interupt is generated      %
%                                                                                              %

EXT_INT = DSP0IRQ;

%                                                                                              %
% Assign the two data busses thier respective tristate signals                                 %
%                                                                                              %

HD[7..0]  = hdnode[7..0];
SD[15..0] = sdnode[15..0];

END;
```